KERNFORSCHUNGSANLAGE JÜLICH GmbH

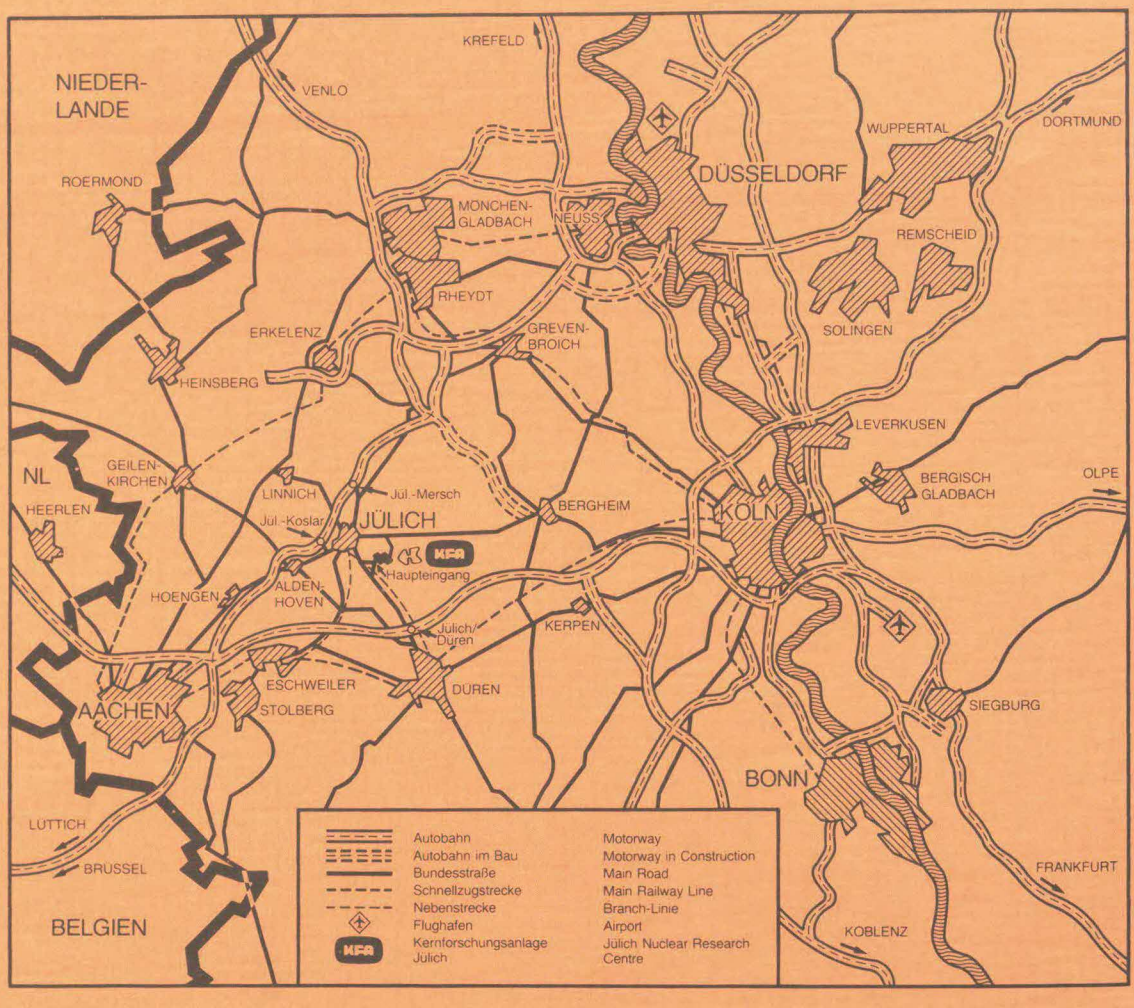
Zentralinstitut für Angewandte Mathematik

Pipeline-Scheduling und parallele Graphenalgorithmen

von

J. Tappe

Jül - Spez - 284
Januar 1985
ISSN 0343-7639



Als Manuskript gedruckt

Berichte der Kernforschungsanlage Jülich – Nr. 284

Zentralinstitut für Angewandte Mathematik Jül - Spez - 284

Zu beziehen durch: ZENTRALBIBLIOTHEK der Kernforschungsanlage Jülich GmbH

Postfach 1913 · D-5170 Jülich (Bundesrepublik Deutschland)

Telefon: 02461/610 · Telex: 833556-0 kf d

Pipeline-Scheduling und parallele Graphenalgorithmen

von

J. Tappe*

*Lehrstuhl B für Mathematik der RWTH Aachen

Für die großzügige Unterstützung beim Verfassen der vorliegenden Arbeit, die in Zusammenarbeit mit dem Zentralinstitut für Angewandte Mathematik der Kernforschungsanlage Jülich entstand, möchte ich dem Direktor dieses Instituts, Herrn Dr. F. Hoßfeld herzlich danken.

Pipeline-Scheduling und parallele Graphenalgorithmen

Jürgen Tappe

1. Einleitung

Eine effektive Methode zur Steigerung von Rechnerleistungen ist das in Abschnitt 2 beschriebene lineare Pipelineprinzip. Hierbei wird durch Segmentierung eines Rechenvorgangs erreicht, daß nach Ablauf einer Startzeit pro Maschinenzeittakt ein Resultat geliefert wird. Den nicht zu übersehenden Vorteilen des linearen Pipelinings steht der Nachteil gegenüber, daß die Nutzung der Pipelinesegmente häufig nur auf wenige Rechenabläufe beschränkt ist. Ein variablerer Einsatz der Segmente (z.B. Feedback) führt zu einem allgemeineren Konzept, dem statischen Pipelining (Abschnitt 2).

Das Problem der Steuerung dieser Pipelines wurde in [2] graphentheoretisch formuliert. Für statische Pipelines besteht das Hauptproblem darin, im zugehörigen bewerteten Zustandsgraphen Kreise minimaler Durchschnittslänge MAL (minimale durchschnittliche Verzögerung der Pipeline, engl.: minimal average latency) zu ermitteln. In der vorliegenden Arbeit werden Berechnungsmethoden von MAL, zugehöriger Kreise und zugehöriger Startwege diskutiert (Abschnitt 3, 4 und 5). Der Anhang (Abschnitt 6) enthält eine kurze Beschreibung mehrerer APL Programme, die sich für die Steuerung statischer Pipelines mit wenigen Zuständen eignen.

2. Statische Pipelines

Das in dieser Arbeit betrachtete Pipelinekonzept ist in [6] ausführlich beschrieben, daher soll hier nur eine kurze Betrachtung erfolgen. Pipelines können in Rechnern überall dort gewinnbringend eingesetzt werden, wo bestimmte Rechenschritte häufig wiederholt werden. Statt einer Funktionseinheit, die die gewünschte Rechnung in mehreren Zeittakten durchführt, wird die Ausführung auf zeitgleich und unabhängig voneinander arbeitende Segmente unterteilt. Das hat zur Folge, daß zwar der einzelne Rechenschritt dieselbe Ausführungszeit hat, jedoch mehrere Operationen im Abstand eines kurzen Zeittaktes überlappt bearbeitet werden. Der zeitliche Ablauf von 6 Rechnungen A, B, C, D, E, F in einer Pipeline mit 4 Segmenten läßt sich am folgenden Diagramm verdeutlichen:

Zeittakte

Segmente	0	1	2	3	4	5	6	7	8
S_0	A	B	C	D	E	F			
S_1		A	B	C	D	E	F		
S_2			A	B	C	D	E	F	
S_3				A	B	C	D	E	F

Man erkennt, daß 6 gleichartige Rechenschritte, die jeweils 4 Zeittakte benötigen, durch die Segmentierung in nur 9 Zeittakten abgeschlossen sind.

Die oben beschriebene Methode wollen wir im folgenden das lineare Pipelinekonzept nennen. Verlangt man, daß die Segmente eines Rechners nicht nur in linearer Folge für eine stets festgelegte, meist elementare Operation Verwendung finden soll, so führt dies zu der folgenden Verallgemeinerung, der sog. statischen Pipeline: Gegeben seien Einheiten (Segmente) $S_0,S_1,\ldots,S_{m-1}$ eines Prozessors, die während jeden Zeittaktes neu belegt werden dürfen und die wir, bezogen auf eine feste Rechenoperation eine statische Pipeline nennen. Eine (k Zeittakte dauernde) Rechenoperation R ist eine Belegungsfolge der Segmente in k aufeinander folgenden Zeittakten $0,1,2,\ldots,k-1$. Diese Situation läßt sich am einfachsten durch eine sogenannte Reservierungstabelle darstellen, d.h. eine m x k-Matrix $R_o=(r^o_{ij})$ mit $r^o_{ij}=1$ oder 0, je nachdem, ob S_i im Zeittakt j belegt ist oder nicht. Den Trivialfall der Nullmatrix wollen wir ausschließen. Für $i=1,2,\ldots,$ k-1 sei R_i die Matrix, deren Spalten $0,1,2,\ldots,i-1$ verschwinden und deren Spalten $i,i+1,\ldots,k-1$ mit den Spalten $0,1,\ldots,k-i-1$ von R_0 übereinstimmen. Offensichtlich läßt die statische Pipeline nach Initiierung von R zur Zeit o eine weitere Initiierung von R zur Zeit t genau dann zu, wenn in keiner Position $r^o_{ij} = r^t_{ij} = 1$ gilt. Als Kollisionsvektor der Pipeline bezeichnen wir den binären Vektor $KV=(v_0,v_1,\ldots,v_{k-1})$, wobei v_t genau dann verschwindet, falls R_o und R_t die obige Bedingung erfüllen. Offensichtlich ist $v_0=1$. Als (Steuerungs-)Zustände der Pipeline wollen wir im folgenden die binären Vektoren $E=(e_0,e_1,\ldots,e_{k-1})$ bezeichnen, die zu endlich vielen Initiierungen von R gehören und deren Komponenten folgendes angeben: erfolgte die letzte Initiierung zu Zeit T, so verschwindet e_i genau dann, wenn eine erneute Initiierung zur Zeit T+i möglich ist. Folgendes ist leicht einzusehen:

2.1 Bemerkung

(i) KV ist ein Zustand.

(ii) Zustände entstehen aus gegebenen Zuständen durch mit Nullen auffüllende Linksshifts bis eine Null in Position 0 erscheint und anschließender logischer Oder-Verkünpfung mit KV.

Wir definieren nun einen bewerteten Digraphen G, den Zustandsgraphen der Pipeline (vgl. [2], [6]): Die Ecken von G seien die Zustände der Pipeline. Ein Pfeil weist genau dann von E_1 nach E_2, wenn E_2 aus E_1 gemäß 2.1(ii) entsteht. Die Länge $l(E_1,E_2)$ des Pfeils (E_1,E_2) ist die Minimalzahl der Linksshifts von E_1, so daß gemäß 2.1(ii) der Zustand E_2 entsteht.

2.2 Interpretation.

Wege in G beginnend im Punkte KV entsprechen Steuerungsfolgen der Pipeline. Der Start in KV entspricht dem Start der Pipeline zum Zeitpunkt 0 (erste Initiierung von R). Dem Übergang von einem zu einem benachbarten Zustand längs eines Pfeils entspricht eine erneute Initiierung von R. Die Länge des Pfeils gibt an, wieviele Zeittakte später diese erfolgt.

Aus der Definition von G leitet man ohne Schwierigkeiten folgende Eigenschaften ab:

2.3 Bemerkung.

(i) G ist endlich mit höchstens 2^m Ecken, dabei sei m die Zahl der Nullen in KV, die vor der letzten Eins von KV stehen; Beispiel: KV = (10010100), m=3.

(ii) G ist ein stark zusammenhängender Graph, der Zyklen enthalten kann. Jede von KV verschiedene Ecke besitzt einen Pfeil der nach KV weist.

(iii) es sei m wie oben definiert. Dann hat jede Ecke von G höchstens m+1 direkte Nachfolger.

Alle Steuerungsfolgen haben eines gemeinsam, nämlich die Startinitiierung, die wir in der Regel nie mitzählen wollen. Es sei nun $W=(E_0=KV,E_1,E_2,\ldots E_t)$ ein beliebiger Weg in G. Von Bedeutung sind die folgenden Größen:

(i) t, die Anzahl der Initiierungen von R (ohne die Startinitiierung),

(ii) $l(W)=\sum_{i=0}^{t-1} l(E_i,E_{i+1})$, die Gesamtverzögerung der Steuerungsfolge

(iii) $l(W)/t$, die mittlere Verzögerung von W.

Bricht man nach dem Durchlaufen von W die Berechnung ab, so wurden in $l(W)+k$ Zeittakten insgesamt t+1 (incl. Startinitiierung) Operationen vom Typ R ausgeführt. Für die Betrachtung optimaler Steuerungsfolgen gehen wir von einer großen Zahl von Operationen aus. Da G endlich ist besteht W für große Werte t (bis auf vernachlässigbare Endstücke) aus einem geschlossenen Weg; eine sinnvolle Steuerung kann ohnehin nicht unendlich sein. Jeder geschlossene Weg ist aus (einfachen) Kreisen zusammengesetzt. Nach einem Resultat von Shar (vgl. [7]) ist die mittlere Verzögerung einer geschlossenen Linie größer oder gleich dem Minimum der mittleren Verzögerungen aller beteiligter Kreise. Mit MAL (minimum average latency) bezeichnen wir das Minimum der mittleren Verzögerungen gebildet über alle Kreise von G, die zugehörigen Kreise nennen wir MAL-Kreise. Eine sinnvolle Steuerung der Pipeline besteht daher aus einem Weg in G, der von KV in einen MAL-Kreis führt, d.h. zu studieren sind Folgen benachbarter Ecken $E_0=KV,E_1,E_2,\ldots,E_r=E_t, E_{r+1},E_{r+2},\ldots,E_t$ mit minimalem Wert

$$\frac{1}{t-r}\sum_{i=r}^{t-1} l(E_i,E_{i+1}).$$

Zum Abschluß dieses Abschnitts sei noch auf einige Bemerkungen hingewiesen, die sich leicht aus der Definition des Zustandsgraphen ergeben. Nullen am Ende eines Kollisionsvektors können weggelassen werden, ohne daß sich der Zustandsgraph ändert. Zu jedem binären Vektor, dessen erste Komponente den Wert 1 hat, gibt es eine Reservierungstabelle, die den vorgegebenen Vektor als Kollisionsvektor hat, d.h. KV kann jede denkbare Form haben. (Man wähle hierzu als R_0 eine passende Einheitsmatrix bei der die erste Spalte durch KV ersetzt wird.) Das Beispiel KV=(1,0,0,...,0,1) der Dimension k liefert einen Graphen mit 2^{k-2} Ecken, d.h. die in 2.3(i) angegebene Schranke ist scharf. Die maximale Ausgangsvalenz hingegen beträgt nur k-1. Es treten also Fälle auf in denen die Adjazenzmatrizen von Zustandsgraphen dünn besetzt sind.

Das lineare Pipelinekonzept läßt sich anhand des nachstehenden Resultats vielfältig charakterisieren:

2.4 Bemerkung.

Es sei KV der Kollisionsvektor einer statischen Pipeline und MAL die minimale mittlere Verzögerung. Folgende Aussagen sind äquivalent

(i) MAL = 1

(ii) MAL < 2

(iii) KV hat die Form (1,0,0,...,0)

(iv) Der Zustandsgraph G besteht nur aus einem Zyklus der Länge 1.

Beweis: Die Aussagen (i) → (ii), (iii) → (iv) und (iv) → (i) sind trivial. Es sei MAL < 2. Folglich hat G einen Kreis, der Pfeile der Länge 1 enthält und damit auch Zustände der Form Z=(1,0,*,*,...,*). Nimmt * in mindestens einem Fall den Wert 1 an, so folgt ausgehend von Z auf jede konsekutive Folge von r Pfeilen der Länge 1 ein Pfeil der Mindestlänge r+2, was MAL ≧ 2 nach sich zieht. Also ergibt sich Z=KV=(1,0,0,...,0).

Wünschenswert ist nun eine Methode, die MAL und optimale Steuerungsfolgen ohne großen Aufwand bestimmt. Ein erster Ansatz, der in manchen Fällen gute Resultate liefert beruht auf einigen, zum Teil trivialen Feststellungen. Es seien $a_1,a_2,\ldots,a_r$, die Längen der Pfeile $P_1,P_2,\ldots,P_r$ von KV zu seinen unmittelbaren Nachfolgern $E_1,E_2,\ldots E_r$ und $b_1,b_2,\ldots b_r$ die Längen der kürzesten Pfeile die von $E_1,\ldots,E_r$ ausgehen. Mit M bezeichnen wir das Minimum der Mittelwerte $(a_1+b_i)/2$. Da jeder Zustand E aus KV durch Hinzufügen zusätzlicher Einsen entsteht, ist jeder Pfeil P aus E einem Pfeil P_i aus KV zuzuordnen, so daß $l(P) \geq l(P_i)$ gilt und der durch P gegebene Nachfolgezustand von E aus E_i durch Einfügen von Einsen entsteht. Damit ist klar, daß auf P ein Pfeil der Länge $\geq b_i$ folgen muß. Wir betrachten nun einen Kreis K in G, der durch die Pfeile $Q_1,Q_2,\ldots Q_s$ gegeben ist. Die zugehörigen Pfeile aus KV bezeichnen wir mit $P(Q_j)$, die zugehörigen Werte a_j, b_j mit $a(Q_j)$, $b(Q_j)$. Es gilt demnach $l(Q_i) \geq l(P(Q_i))=a(Q_i)$ und nach obiger Überlegung folgt $l(Q_{i+1}) \geq b(Q_i)$. Somit folgt für die Länge des Kreises

$$l(K)=\sum_{i=1}^{s} l(Q_i) \geq \sum_{i=1}^{s} a(Q_i) \text{ und } l(K) \geq \sum_{i=1}^{s} b(Q_i)$$

Daraus ergibt sich $l(K) \geq s \cdot M$ und wir erhalten:

2.5 Bemerkung. Es gilt $M \leq MAL$.

Eine recht einfache Möglichkeit, an eine Steuerungsfolge zu kommen, ergibt sich aus folgender Betrachtung des Kollisionsvektors $KV=(V_0,V_1,\ldots,V_{k-1})$. Gilt für eine natürliche Zahl m, daß $V_m,V_{2m},V_{3m},\ldots$ sämtlich verschwinden, so liefert die konstante Verzögerungsfolge mit dem Wert m eine Steuerungsfolge (d.h. von KV aus werden in G nur Pfeile der Länge m durchlaufen). Aus der Definition von

G läßt sich leicht ableiten, daß der zugehörige Weg in G die Form E_0=KV, $E_1,E_2,\ldots E_t,E_t$ hat, mit $l(E_i,E_{i+1})=l(E_t,E_t)=m$, d.h. der Weg führt in einen m-Zyklus.

Die beiden obigen Betrachtungen lassen sich in manchen Beispielen gewinnbringend kombinieren, z.B. im Falle KV=(11001100001101). Der Tabelle in § 3 entnehmen wir, daß die optimale Steuerung unter 3172 Folgen zu suchen ist. Man erkennt jedoch sofort, daß hier M=3 und damit 3 ≦ MAL gilt. Andererseits ist an KV abzulesen, daß der Graph die konstante Verzögerungsfolge mit dem Wert 3 gestattet. Diese Steuerung erweist sich sogar im Sinne der in § 4 angegebenen Bewertungskriterien für die Startwege als optimal. Es ist jedoch durch Beispiele leicht zu belegen, daß der obige Ansatz keine Allgemeingültigkeit hat, d.h. es sind aufwendigere Methoden nötig, um für beliebige Kollisionsvektoren die optimalen Steuerungen zu ermitteln.

Zu erwähnen ist noch die sog. Greedy-Strategie (vgl. [3]), bei der ausgehend von KV stets die kürzeste Verzögerung (Pfeillänge) gewählt wird. Leider führt dieses Verfahren in vielen Fällen nicht zu Steuerungen mit MAL-Kreisen.

3. Die lexikographische Suchmethode

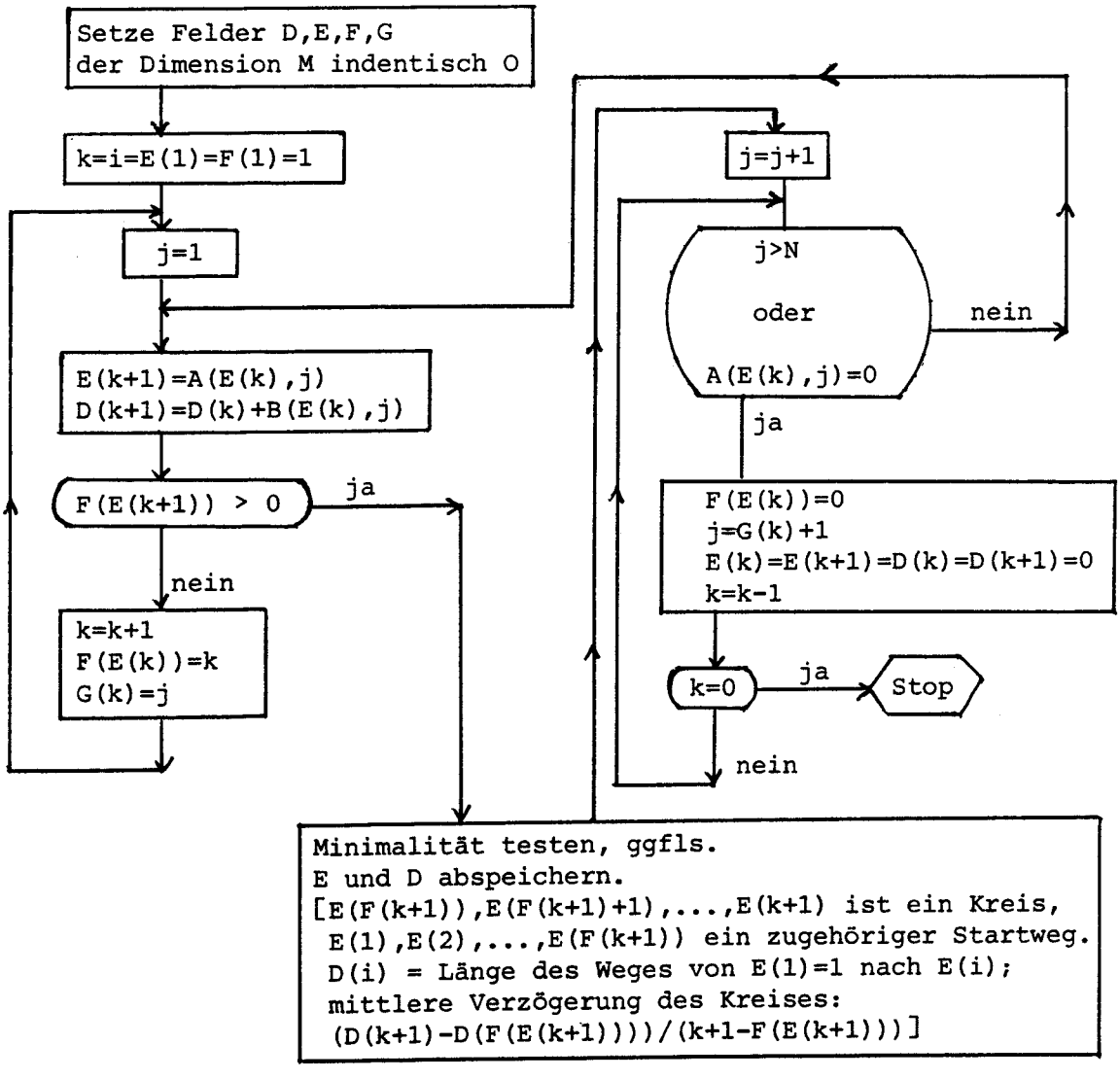
Wir betrachten in diesem Abschnitt einen bewerteten Digraphen G mit einer ausgezeichneten Ecke KV. Gesucht sind sowohl die MAL-Kreise, als auch Wege von KV zu vorgegebenen MAL-Kreisen (vgl. § 2). In [8] wird folgende Methode vorgeschlagen: Ausgehend von jeder Ecke wurden die Folgen benachbarter Ecken betrachtet, tritt eine Ecke ungleich der aktuellen Ausgangsecke doppelt auf, so ist die aktuelle Folge zu streichen. Auf diese Weise entstehen sämtliche Kreise. Die Startwege von KV zu den optimalen Kreisen werden separat behandelt.

Beschränkt man sich bei der obigen Konstruktion auf einen Ausgangspunkt, nämlich KV und ermittelt Eckenfolgen nach einem lexikographischen Prinzip, so erhält man in einem Bruchteil des Aufwandes und ohne unkontrolliertes Anwachsen des Speicherbedarfes sämtliche Kreise inklusive aller doppelpunktfreien Startwege. Wir identifizieren dazu die Ecken von G mit den natürlichen Zahlen 1,2, 3,..,a, wobei 1 die Nummer von KV sei. Gesucht sind alle Folgen benachbarter Ecken $1=e_1,e_2,e_3,\ldots,e_t$ mit $e_i \neq e_j$ für alle $i\neq j, i\leq t-1, j\leq t-1$ und $e_t=e_i$ für ein $i\leq t-1$. Diese Folgen können durch die Anordnung der natürlichen Zahlen lexikographisch angeordnet werden. Das nachstehende Diagramm 3.1 zeigt ein Programm, das die obigen Folgen in der lexikographischen Reihenfolge berechnet (jede Folge wird genau einmal durchlaufen). Als Eingabe dient eine Nachfolgertabelle

Diagramm 3.1

Eingabe: N ≧ maximale Ausgangsvalenz
M = Anzahl der Ecken
A(M,N) = Nachfolgerliste des Graphen, die Zeilen sind angeordnet und mit Nullen aufgefüllt
B(M,N) = Matrix der zugehörigen Pfeillängen

Ausgabe: optimale Eckenfolgen der Form E(1),E(2),..., die jeweils einen Startweg und einen MAL-Kreis darstellen



A des Graphen G, d.h. die i-te Zeile von A enthält die Nachfolger der Ecke i in ihrer natürlichen Reihenfolge, überzählige Tabellenpositionen werden mit Nullen besetzt. Eine parallele Tabelle B enthält die zugehörigen Pfeillängen.

Die Vorteile des oben angegebenen Verfahrens liegen sicher darin, daß Kreise und Startwege simultan und mit einem Minimum an Speicherbedarf ermittelt werden. Trotz der Reduzierung des Aufwandes gegenüber [8] bleibt jedoch zu bemerken, daß in größeren Beispielen die Zahl der lexikographisch geordneten Folgen sehr groß werden kann; Beispiele:

Kollisionsvektor	Anzahl der Ecken von G	Anzahl der Folgen
(100001)	16	343
(110001001011101)	18	1995
(11001100001101)	10	3172
(110001000111001)	20	4957
(110011010100101)	12	10274
(10010010001)	11	2068
(110000110000111001)	24	> 52000

Um den hohen Aufwand bei der Suche von MAL-Kreisen zu reduzieren, kann es sich lohnen, gewisse Pfeile des Graphen zu streichen. Wir betrachten hierzu Pfeile $(e_1,e_2),(e_2,e_3),(e_3,e_4),(e_1,e_3),(e_1,e_4)$ von G. Ferner sei M eine untere Schranke für MAL (man kann für M das in 2.5 betrachtete Minimum bzw. nach 2.4 M=2 wählen). Gilt nun $M \geq l(e_1,e_2)+l(e_2,e_3)-l(e_1,e_3)$, so ändert sich durch das Weglassen des Pfeils (e_1,e_3) der Wert von MAL nicht; dasselbe gilt für (e_1,e_4), falls $2M \geq l(e_1,l_2)+l(e_2,e_3)+l(e_3,e_4)-l(e_1,e_4)$. Um dies zu beweisen, mache man sich klar, daß man in jedem MAL-Kreis, der (e_1,e_3) bzw. (e_1,e_4) enthält, den gestrichenen Pfeil durch die Komposition von $(e_1,e_2),(e_2,e_3)$ bzw. (e_1,e_2), $(e_2,e_3),(e_3,e_4)$ ersetzen kann. Es können dabei jedoch gewisse, wenn auch nicht alle MAL-Kreise verlorengehen. Ähnliche Reduktionskriterien lassen sich auch für längere Pfeilketten leicht herleiten.

4. Zur Bewertung der Startwege

Bei der Betrachtung zur Steuerung statischer Pipelines liegt der Schwerpunkt auf der Bestimmung der MAL-Kreise. Ist die Zahl der zu wiederholenden Operationen groß gegen die Zahl der Ecken des Zustandsgraphen, so spielt die Wahl der Startwege nur eine untergeordnete Rolle. Soll die Steuerung jedoch auch häufiger für eine kleine Anzahl von Operationen benutzt werden, so kann die Wahl des Startweges, der vom Anfangspunkt KV in einen MAL-Kreis führt, durchaus von Bedeutung sein. In [8] werden die Startwege anhand der mittleren Verzögerung bewertet. Damit bleiben aber die leeren Startwege zu MAL-Kreisen durch KV ohne Bewertung. Außerdem wird man bei einen Beispiel mit MAL=5 in der Regel einen Startweg mit den Verzögerungen 2, 3 und 2 einem Startweg vorziehen, der nur aus einem Pfeil der Länge 2 besteht, obgleich dessen mittlere Verzögerung kleiner ist.

Als Alternative wollen wir im folgenden zwei Bewertungsmöglichkeiten angeben, von denen die erste auch die Struktur der MAL-Kreise berücksichtigt. Dazu wählen wir zunächst eine natürliche Zahl m und betrachten zwei von KV ausgehende und in MAL-Kreise mündende Steuerungsfolgen S_1 und S_2. Mit a_i sei der Zeitpunkt der i-ten Initiierung der zugrundeliegenden Operation bei Steuerung anhand von S_1 bezeichnet, analog seien $b_1,b_2,b_3,\ldots$ für S_2 definiert. Man ziehe nun S_1 der Folge S_2 vor, sofern

$$\sum_{i=1}^{m} a_i < \sum_{i=1}^{m} b_i$$

gilt. Man geht dabei von der Vorstellung aus, daß häufig bis zu m+1 Initiierungen (incl. Startinitiierung) von der Pipeline geleistet werden sollen und daß die auftretenden Anzahlen in etwa gleich verteilt sind. Wählen wir die Folge S_1, so ist damit die Größe

$$\frac{1}{m} \cdot \sum_{i=1}^{m} (b_i - a_i)$$

der Erwartungswert des Zeitgewinns. Im nachfolgenden Beispiel sind die Steuerungsfolgen $S_1,S_2,\ldots,S_5$ gegeben durch die Folge der Verzögerungen (Pfeillängen) ausgehend vom Zustand KV=(100001). Die unterstrichenen Teilfolgen kennzeichnen die MAL-Kreise. Die gewählte Reihenfolge entspricht obiger Bewertung für m=16.

KV = (100001), MAL = 2

S_1: 1 1 1 1 6

S_2: 1 1 2 4 2 1 1 2

S_3: 1 1 2 4 2 1 3 3 1 2 1

S_4: 2 2 2

S_5: 3 1 2 1 3 3

Die Definition des Auswahlkriteriums macht klar, daß die Bewertung der Folgen von der Größe m abhängt, insbesondere für kleine m. Um dies genauer zu analysieren, betrachten wir erneut die Steuerungsfolgen S_1 und S_2 mit den zugehörigen Zeitfolgen $(a_i)_{i\in N}, (b_i)_{i\in N}$. Ferner sei $c_i=a_i-a_{i-1}, d_i=b_i-b_{i-1}$ für $i\geq 2$ und $c_1=a_1, d_1=b_1$, d.h. $(c_i)_{i\in N}, (d_i)_{i\in N}$ sind die zugehörigen Verzögerungsfolgen. Es sei t das kleinste gemeinsame Vielfache der Schrittzahlen der in S_1 und S_2 enthaltenen MAL-Kreise und g die die kleinste nicht negative ganze Zahl mit der Eigenschaft, daß sich ab Steuerungsschritt g+1 beide Steuerungsfolgen innerhalb ihrer MAL-Kreise befinden. Alle Werte $m\geq g$ lassen sich nun wie folgt darstellen: $m=g+s\cdot t+1$ mit $s\geq 0, 0\leq r\leq t-1$. Für alle $u\geq g$ gilt offenbar $b_u-a_u=b_{u+t}-a_{u+t}, c_u=c_{u+t}$, $d_u=d_{u+t}$ und wir erhalten:

$$\sum_{i=1}^{m}(b_i-a_i)=\underbrace{\sum_{j=1}^{g}(b_j-a_j)}_{A}+s\cdot\underbrace{\left[t(b_g-a_g)+\sum_{i=g+1}^{g+t}(g+t-i+1)(d_i-c_i)\right]}_{B}+\underbrace{r\cdot(b_g-a_g)+\sum_{k=g+1}^{g+r}(g+r-k+1)(d_k-c_k)}_{C(r)}$$

$$= A+s\cdot B+C(r)$$

Man erkennt, daß A und B konstant sind, C(r) ist beschränkt. Gilt B>0, so ist damit

$$\sum_{i=1}^{m}(b_i-a_i)$$

für fast alle m positiv. Für hinreichend große m ist somit die Bewertung von S_1 und S_2 unabhängig von m; ausschlaggebend ist dann die Zeit für die ersten g Schritte und die darauf folgende Verteilung der Verzögerungen innerhalb der MAL-Kreise. Falls B=0 ist, so sind hingegen nur die Werte m=1,2,3,...,g+t-1 maßgebend, denn für m, m'≥g gilt

$$\sum_{i=1}^{m} (b_i - a_i) = \sum_{i=1}^{m'} (b_i - a_i),$$

sofern $m \equiv m' \bmod t$. In obigen Beispiel zeigt sich, daß die Folgen S_4 und S_5 für $m \equiv 0$ oder 4 mod 5 gleich bewertet werden, hingegen ist für $m \equiv 1,2$ oder 3 mod 5 die Folge S_4 der Folge S_5 vorzuziehen. Der Erwartungswert des Zeitgewinns von S_1 im Vergleich zu S_2 konvergiert für wachsende m gegen B/t, was sich unmittelbar aus $\Sigma(b_i - a_i) = A + sB + C$ und $m = g + s \cdot t + r$ ergibt.

Die obige Bewertung geht davon aus, daß (wie bei der lexikographischen Suche) alle Steuerungsfolgen mit MAL-Kreisen berechnet werden. Bei den Matrixmethoden in § 5 ist diese Situation nicht unbedingt gegeben. Hier wird zunächst nur MAL ermittelt, ferner sind Weglängen und Schrittzahlen bekannt. Die MAL-Kreise und die Startwege selbst werden erst nach der Wahl der Startecke eines MAL-Kreises ermittelt, etwa durch ein Backtrackingverfahren. Hier ist eine a-priori-Bewertung nötig, die naturgemäß mit geringerer Genauigkeit arbeitet. Eine Möglichkeit stellt sich wie folgt dar: Es sei n die Anzahl der Zustände in G. Wir bewerten eine Steuerungsfolge mit MAL-Kreis, deren Startweg r Initiierungen und eine Länge von d hat mit $d + (n-r) \cdot MAL$. Nachteilig, aber in vielen Fällen unvermeidlich ist, daß die Verteilung der Verzögerungen im Startweg und im MAL-Kreis nicht berücksichtigt wird. Im obigen Beispiel (KV=(100001)) ergibt sich nun folgende Reihenfolge: S_2, S_3, S_1 und S_4, S_5, obgleich die Zeitfolge von S_1 (dies ist die Greedy-Strategie) im Vergleich zu allen anderen stets die besseren Werte liefert. Im allgemeinen führt die Greedy-Strategie, wie schon in § 2 bemerkt, nicht immer zu MAL-Kreisen. Im Beispiel KV=(1010101101001) hingegen führt die Greedy-Strategie zu einem MAL-Kreis, sie ist aber nach keinem der obigen Bewertungskriterien optimal (die Greedy-Strategie hat die Verzögerungsfolge 1, 10, 3, 5; besser ist die Folge 3, 5, 3, 5).

5. Matrixalgorithmen zur Bestimmung von MAL

In diesem Paragraph sei G der Zustandsgraph einer statischen Pipeline mit n Zuständen (Ecken) $E_1, E_2, \ldots, E_n$ und der Pfeilbewertung 1. Bei aller Geradlinigkeit des lexikographischen Algorithmus aus § 3 ist jedoch einschränkend festzustellen, daß für die Zahl der Rechenschritte keine Schranke angegeben werden kann, die polynomial in n ist. Die hier betrachteten Matrixmethoden benötigen zwar mehr Speicherplatz, sind dafür aber polynomial und parallelisierbar. Wir

betrachten die folgende n x n-Matrix $A=(a_{ij})$ mit $a_{ij}=l(E_i,E_j)$, falls in G ein Pfeil von E_i nach E_j weist, andernfalls sei $a_{ij}=\infty$. Für $k\varepsilon N$ seien die Matrizen $A^{(k)}=(a_{ij}^{(k)})$ gegeben, wobei $a_{ij}^{(k)}$ die minimale Länge eines Weges mit genau k Pfeilen von E_i nach E_j angibt; wenn kein solcher Weg existiert, sei $a_{ij}^{(k)}=\infty$. Offensichtlich gilt $A=A^{(1)}$ und

$$a_{ij}^{(s+t)} = \operatorname{Min}_{r} \{a_{ir}^{(s)}+a_{rj}^{(t)}\},$$

d.h. die Matrizen $A^{(k)}$ sind die Potenzen von A bezüglich des in der Graphentheorie üblichen Min-Add-Matrixproduktes. Die Resultate in [4], [5] zeigen, daß ein Parallelrechner mit hinreichend vielen Prozessoren die Matrizen $A^{(1)}$, $A^{(2)},\ldots,A^{(n)}$ in $O(\log^2 n)$ Schritten berechnen kann. Die Bedeutung dieser Matrizen für die Steuerung der Pipeline liegen darin, daß (unter der Voraussetzung E_1=KV) die Werte $a_{ij}^{(k)}$ kürzeste Startweglängen angeben bei vorgegebener Zahl k von Initiierungen; last not least gilt

$$MAL=\operatorname{Min}_{i,k}\left\{\frac{a_{ii}^{(k)}}{k}\right\},\ i,k\varepsilon\{1,2,\ldots,n\}.$$

Dabei ist zu beachten, daß nicht in jedem Fall aus $MAL=a_{ii}^{(k)}/k$ folgt, daß in E_i ein MAL-Kreis mit k Initiierung existiert. Es kann vielmehr die Situation vorliegen, daß eine geschlossene Linie mit Durchschnittsverzögerung MAL gegeben ist, die sich aus MAL-Kreisen zusammensetzt. Bestimmt man die Matrizen $A^{(1)}$, $A^{(2)},A^{(3)},\ldots$ in Serie, so kann man den obigen Effekt zumindest etwas einschränken, indem man nach Auswertung der Diagonalelemente von $A^{(k)}$ vor der Berechnung von $A^{(k+1)}$ die Diagonalelemente gleich ∞ setzt. Ist neben $MAL=a_{ii}^{(k)}/k$ außerdem k minimal gewählt, so entspricht $a_{ii}^{(k)}$ in jedem Fall einem MAL-Kreis. Mit Hilfe der in § 4 angegebenen a-priori-Bewertung läßt sich aus den Werten $a_{ij}^{(k)}$ eine geeignete Ecke E_j finden, die Endpunkt eines Startwegs und Startpunkt eines MAL-Kreises ist. Den Startweg findet man durch folgendes Backtracking-Verfahren: $E_j=F_k,F_{k-1},\ldots,F_s$ seien definiert. Wähle $F_{s-1}\varepsilon\{E_1,E_2,\ldots,E_n\}$, so daß $a_{1u}^{(s-1)}+a_{uw}=a_{1w}^{(s)}$ mit $F_s=E_w$ und $F_{s-1}=E_u$; dies ergibt einen Weg der Länge a_{1j}^{k} mit k Initiierungen von KV nach E_j. Analog ermittelt man einen MAL-Kreis durch E_j. Ohne die Details genauer auszuführen, sei noch darauf hingewiesen, daß durch Verdoppelung des Speicherplatzes beim Backtracking Zeit gespart werden kann. Beim seriellen Berechnen der Matrizen $A^{(k)}$ kann man stets die optimalen Vorgänger zu den Werten $a_{ij}^{(k)}$ speichern, so daß die Suche des geeigneten Zustands F_{s-1} oben entfällt, d.h. die Zeitschranke O(kn) reduziert sich auf O(k). Bei der parallelen Berechnung hingegen fallen Zwischenpunkte auf den optimalen We-

gen an, die, sofern sie gespeichert wurden, ein "paralleles Backtracking" in $O(\log k)$ Schritten ermöglichen. Zur Berechnung optimaler Steuerungsfolgen anhand obiger Matrixmethode sei noch bemerkt, daß die Zeitkomplexität in jedem Falle polynomial bzgl. der Zahl n der Zustände ist, d.h. die Methode erweist sich in vielen Fällen auch auf seriellen Rechnern als überlegen. Bei serieller Berechnung, aber auch bei der Benutzung von Vektorrechnern erscheint die Verwendung der Matrix A unangemessen, da in der Regel die Zahl n der Zustände wesentlich größer ist als die Zahl der direkten Nachfolger der Ecken von G (vgl. 2.3). Somit empfiehlt sich hier eine Darstellung des Graphen durch Nachfolgerlisten, wodurch Zeit und Speicherplatz gespart wird. Bei den kürzesten Weglängen konzentriert man sich dann jeweils auf einen festen Punkt E und definiert $l_k(E_i,E)$ als die Länge des kürzesten Weges von E_i nach E mit genau k Initiierungen, d.h. $l_k(E_i,E)=a_{ij}^{(k)}$, falls $E=E_j$. Es gilt nun

$$l_{k+1}(E_i,E) = \min_{N(i)} \{l(E_i,E_{N(i)})+l_k(E_{N(i)},E)\},$$

dabei durchläuft $E_{N(i)}$ die Nachfolger von E_i in G. Durch geeignete Gather- und Vector-Merge-Funktionen lassen sich die Vektoren $[l_k(E_1,E),l_k(E_2,E),\ldots,l_k(E_n,E)]$ gewinnbringend auf einem Vektorrechner bestimmen.

Zum Abschluß dieses Abschnitts kehren wir noch einmal zur obigen Matrix A zurück. Wir nehmen nun an, daß n relativ groß ist und der uns zur Verfügung stehende Rechner nur ein Min-Add-Matrixprodukt zur gleichen Zeit leisten kann. Sind wir zudem nur am Wert von MAL interessiert, ohne zunächst einen MAL-Kreis tatsächlich zu bestimmen, so kann durch die Betrachtung der Folge $A^{(2^k)}$ das nachstehende Resultat interessant sein:

5.1 Satz. Für alle $i=1,2,\ldots,n$ gilt

$$\lim_{k\to\infty} \left\{ \frac{a_{ii}^{(k)}}{k} \right\} = \text{MAL}.$$

Beweis. Für alle i,j sei s_{ij} die kleinste Schrittzahl aller Wege von E_i nach E_j, d.h. $s_{j1}=1$. Mit L bezeichnen wir die maximale Pfeillänge von G. Zu bemerken ist noch, daß der Punkt $E_1=KV$ einen Zyklus besitzt. Es sei E_i eine Ecke, die auf einem MAL-Kreis liegt, der u Pfeile enthält. Für alle hinreichend großen $k\in\mathbb{N}$ erhalten wir nun eindeutig bestimmte, nicht negative ganze Zahlen $t_j(k)$ und $r_j(k)$, wobei $r_j(k) \leq u-1$, so daß folgendes gilt:

$$k=1+s_{1i}+s_{ij}+r_j(k)+t_j(k)\cdot u.$$

Dieser Zerlegung entspricht ein Weg von E_j nach E_j in k Schritten, nämlich ein Schritt von E_j nach E_1=KV, $r_j(k)$-maligem Durchlaufen des Zyklus von KV, s_{1i} Schritte von KV nach E_i, $t_j(k)$-maligem Durchlaufen des MAL-Kreises und der Rückkehr zu E_j in s_{ij} Schritten. Mit Ausnahme der Pfeile im MAL-Kreis schätzen wir die Längen aller Pfeile des obigen Weges mit L nach oben ab und erhalten mit $c(j)=L(1+s_{1i}+s_{ij}+u)$

$$MAL \leq \frac{a_{jj}^{(k)}}{k} \leq \frac{1}{k}[L(1+s_{1i}+s_{ij}+r_j(k))+t_j(k)\cdot u\cdot MAL],$$

also

$$MAL \leq \frac{a_{jj}^{(k)}}{k} \leq \frac{c(j)}{k} + MAL,$$

und wir erhalten $$\lim_{k\to\infty} \frac{a_{jj}^{(k)}}{k} = MAL.$$

Ein günstigeres Konvergenzverhalten gegen MAL als jede der Folgen

$$\frac{a_{ii}^{(k)}}{k}$$

hat die Folge

$$\min_i \frac{a_{ii}^{(k)}}{k};$$

beschleunigend wirkt sich außerdem, wie schon oben angedeutet, die Beschränkung auf die Teilfolge $\{A^{(2^k)}\}$ aus. Ersetzt man in A sämtliche Diagonalelemente mit dem Wert ∞ durch die maximale Pfeillänge in G, so ergibt sich eine Matrix, die zwar nicht mehr den Graphen G beschreibt, die aber ebenfalls nach obigem Prinzip den Wert MAL bei besserer Konvergenz liefert.

6. Anhang

Dieser Abschnitt enthält neben den Listings eine kurze Beschreibung einiger APL-Programme zur Pipelinesteuerung.

Das Programm

KVEKTOR

verlangt nach dem Aufruf die Eingabe einer Reservierungstabelle (0-1-Matrix). Es weist der Variablen

KVEK

den zugehörigen Kollisionsvektor zu.

Das Programm

GRAPH

fordert nach dem Aufruf die Eingabe eines Kollisionsvektors. Der zugehörige Zustandsgraph wird anhand zweier Matrizen A und B ausgegeben. Die von Null verschiedenen Komponenten von A[I;] zeigen die Nachfolger des Zustands I an. Die zugehörigen Pfeillängen sind durch B [I;] gegeben.

Das Programm

WEG

verlangt nach dem Aufruf als Eingabe eine Nachfolgertabelle, die zugehörige Matrix der Pfeillängen (vgl. A und B des Programms GRAPH) und eine obere Schranke für die Pfeillängen. Es werden folgende Ausgabewerte erzeugt: Die Zeilen der Matrix M stellen Eckenfolgen optimaler Wege dar. Die zugehörigen Initiierungszeitpunkte sind in der Matrix T gegeben. Die Auswahl der optimalen Wege erfolgt anhand der a-priori-Bewertung aus Abschnitt 4. Der Variablen L wird die Zahl aller durchlaufenen Steuerungsfolgen zugewiesen, KO enthält den Wert MAL, K1 die Bewertung des Startweges. Dem Programm liegt das lexikographische Suchverfahren aus Abschnitt 3 zugrunde.

```
      ∇ KVEKTOR
[1]    'RESERVIERUNGSTABELLE'
[2]    XTAB←⍬
[3]    IN←1
[4]    NN←(⍴XTAB)[2]
[5]    KVEK←NN⍴1
[6]    YTAB←XTAB
[7]   M1:IN←IN+1
[8]    →(IN>NN)/M3
[9]    YTAB←(0 1 ↓YTAB),[2] 0
[10]   ZTAB←YTAB∧XTAB
[11]   →(∨/∨/ZTAB)/M1
[12]   KVEK[IN]←0
[13]   →M1
[14]  M3:→(KVEK[NN]=1)/M2
[15]   KVEK←¯1↓KVEK
[16]   NN←NN-1
[17]   →M3
[18]  M2:'KVEK: KOLLISIONSVEKTOR'
      ∇
```

```
      ∇ NTAB NM
[1]    NM←((1,(⍴NM)[2])⍴⍳(⍴NM)[2]),[1] NM
[2]    NM←(((⍴NM)[1],1)⍴((⍳(⍴NM)[1])-1)),[2] NM
[3]    NM
      ∇
```

```
      ∇ GRAPH
[1]    'K-VEKTOR EINGEBEN'
[2]    K←⍞
[3]    I←0
[4]    N←⍴K
[5]    D←(+/~K)+1
[6]    Z←(1,N)⍴K
[7]    A←B←(1,D)⍴0
[8]    J1←1
[9]   M10:I←I+1
[10]   U←Z[I;]
[11]   T←0
[12]   J←0
[13]  M2:J←J+1
[14]   →(J=N)/M1
[15]   U←(1↓U),0
[16]   →(U[1]=1)/M2
[17]   V←U∨K
[18]   →(T=0)/M6
[19]   T1←1
[20]  M7:→(∧/Z[A[I;T1];]=V)/M2
[21]   T1←T1+1
[22]   →(T1≤T)/M7
[23]  M6:L←1
[24]  M3:→(∧/V=Z[L;])/M4
[25]   L←L+1
[26]   →(L≤J1)/M3
[27]   Z←Z,[1] V
[28]   A←A,[1] 0
[29]   B←B,[1] 0
[30]   J1←J1+1
[31]   T←T+1
[32]   A[I;T]←J1
[33]   B[I;T]←J
[34]   →M2
[35]  M4:T←T+1
[36]   A[I;T]←L
[37]   B[I;T]←J
[38]   →M2
[39]  M1:→(1∈A[I;])/M9
[40]   T←T+1
[41]   A[I;T]←1
[42]   B[I;T]←N
[43]  M9:B[I;⍳T]←B[I;⍋A[I;⍳T]]
[44]   A[I;⍳T]←A[I;⍋A[I;⍳T]]
[45]   →(I≠J1)/M10
[46]   'Z[I;]:  ZUSTAND I'
[47]   'A[I;]:  LISTE DER NACHFOLGER VON I'
[48]   'B[I;]:  LISTE DER UEBERGANGSZEITEN'
      ∇
```

```
      ∇ WEG
[1]    'NACHFOLGERTABELLE'
[2]    A←⍬
[3]    'BOGENLAENGEN'
[4]    B←⍬
[5]    'OBERE SCHRANKE DER BOGENLAENGEN, Z.B. LAENGE DES K-VEKTORS'
[6]    C←⍬
[7]    N←(⍴A)[1]
[8]    N1←(⍴A)[2]
[9]    K0←1+N×C
[10]   F←D←C←E←(N+1)⍴0
[11]   E[1]←F[1]←1
[12]   K←1
[13]   L←0
[14]  M1:J←1
[15]  M7:E[K+1]←A[E[K];J]
[16]   D[K+1]←D[K]+B[E[K];J]
[17]   →(F[E[K+1]]>0)/M2
[18]   K←K+1
[19]   F[E[K]]←K
[20]   G[K]←J
[21]   →M1
[22]  M2:K20←F[E[K+1]]-1
[23]   K30←D[K20+1]
[24]   K00←(D[K+1]-K30)÷(K-K20)
[25]   K10←K30+K00×(N-K20)
[26]   →(K00>K0)/M3
[27]   →(K00=K0)/M4
[28]   M←(1,N+1)⍴E[⍳(K+1)],(N-K)⍴0
[29]   T←(1,N+1)⍴D[⍳(K+1)],(N-K)⍴0
[30]   K0←K00
[31]   K1←K10
[32]   →M3
[33]  M4:→(K10>K1)/M3
[34]   →(K10=K1)/M5
[35]   M←(1,N+1)⍴E[⍳(K+1)],(N-K)⍴0
[36]   T←(1,N+1)⍴D[⍳(K+1)],(N-K)⍴0
[37]   K1←K10
[38]   →M3
[39]  M5:M←M,[1] E[⍳(K+1)],(N-K)⍴0
[40]   T←T,[1] D[⍳(K+1)],(N-K)⍴0
[41]  M3:L←L+1
[42]   J←J+1
[43]  M8:→(J>N1)/M6
[44]   →(A[E[K];J]≠0)/M7
[45]  M6:F[E[K]]←0
[46]   J←1+C[K]
[47]   D[K]←E[K]←0
[48]   K←K-1
[49]   →(K≠0)/M8
[50]   'M: MATRIX DER OPTIMALEN WEGE'
[51]   'T: ZEITPUNKT DER INITIIERUNG'
[52]   'L: ANZAHL ALLER WEGE'
[53]   'K0: MITTLERE BOGENLAENGE DER OPTIMALEN KREISE'
[54]   'K1: BEWERTUNG DES ANFANGS'
      ∇
```

Literaturverzeichnis

[1] C. Berge: Graphs and hypergraphs. North Holland, Amsterdam 1973.

[2] E.S. Davidson: The design and control of pipelined function generators. Proc. Int. IEEE Conf. on Systems, Networks, and Computers, Oaxtepec, Mexico, 19-21, 1971.

[3] E.S. Davidson, A.T. Thomas, L.E. Shar, J.H. Patel: Effective control for pipelined computers. Proc. Spring. COMPCON, IEEE No. 75CH 0920-9C, 181-184, 1975.

[4] E. Dekel, D. Nassimi, S. Sahni: Parallel matrix and graph algorithms. SIAM J. Comput: 10, 657-675, 1981.

[5] F. Hoßfeld: Parallele Algorithmen, Springer, 1983.

[6] P.M. Kogge: The architecture of pipelined Computers, Mc Graw-Hill, 1981.

[7] L.E. Shar: Design and scheduling of statically configured pipelines. Report SU-SEL-72-042, Stanford University, Stanford, California, 1972.

[8] A. Sommer: Scheduling-Strategien für Pipeline-Rechner. Jül-Spez-190, Kernforschungsanlage Jülich, 1983.